Bipolar LSI computing elements usher in new era of digital design

Justin Rattner, Jean-Claude Cornet, and M. E. Hoff, Jr.
Intel Corporation

Electronics, September 5, 1974

Bipolar LSI computing elements usher in new era of digital design

Schottky bipolar chip set that is compatible with standard memories outdoes MOS microprocessors in performance and flexibility—the user microprograms it, and it can be configured to fit many computation and control functions

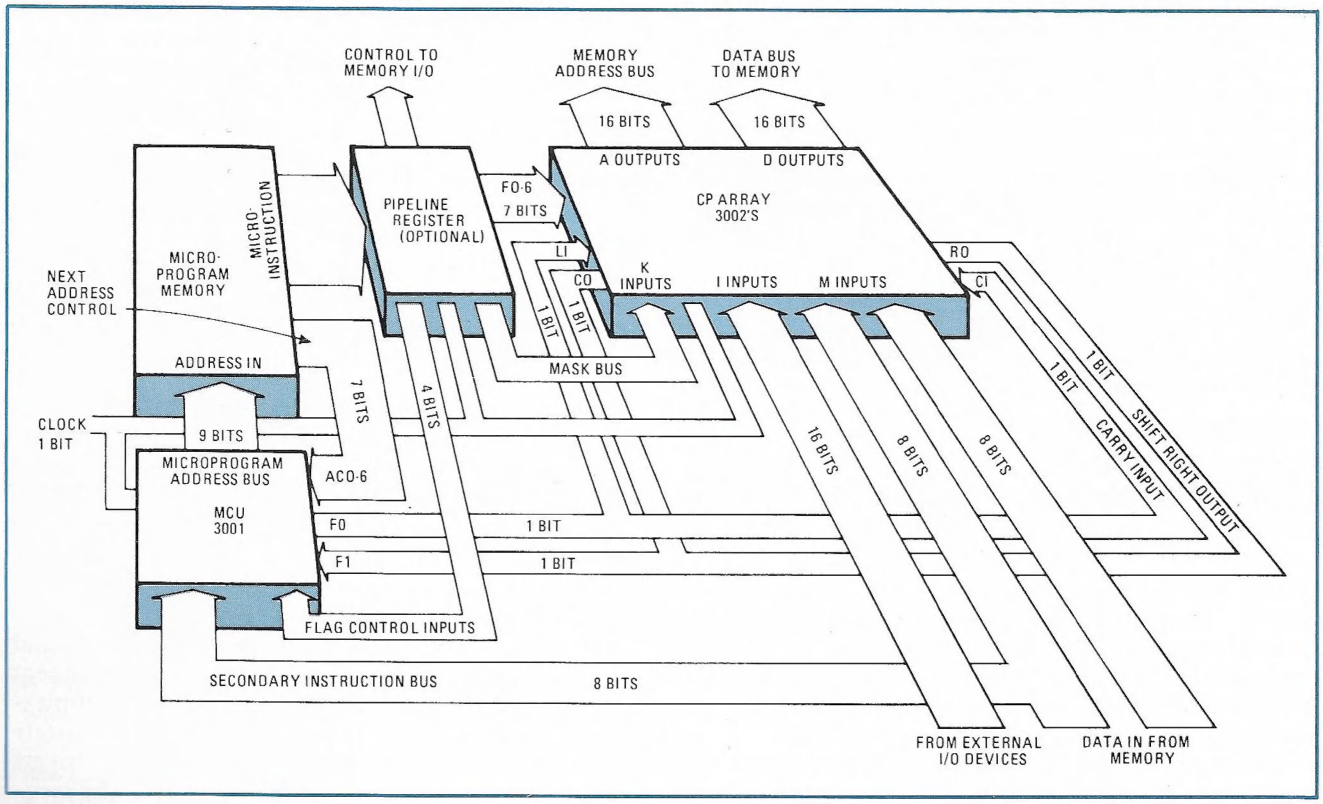
by Justin Rattner, Jean-Claude Cornet, and M.E. Hoff, Jr., *Intel Corp., Santa Clara, Calif.*

☐ Microprocessors have popularized programable LSI technology as a replacement for discrete logic and custom ICs[1]—at least, in the lower-speed applications for which the performance of metal-oxide-semiconductor technology is suited.

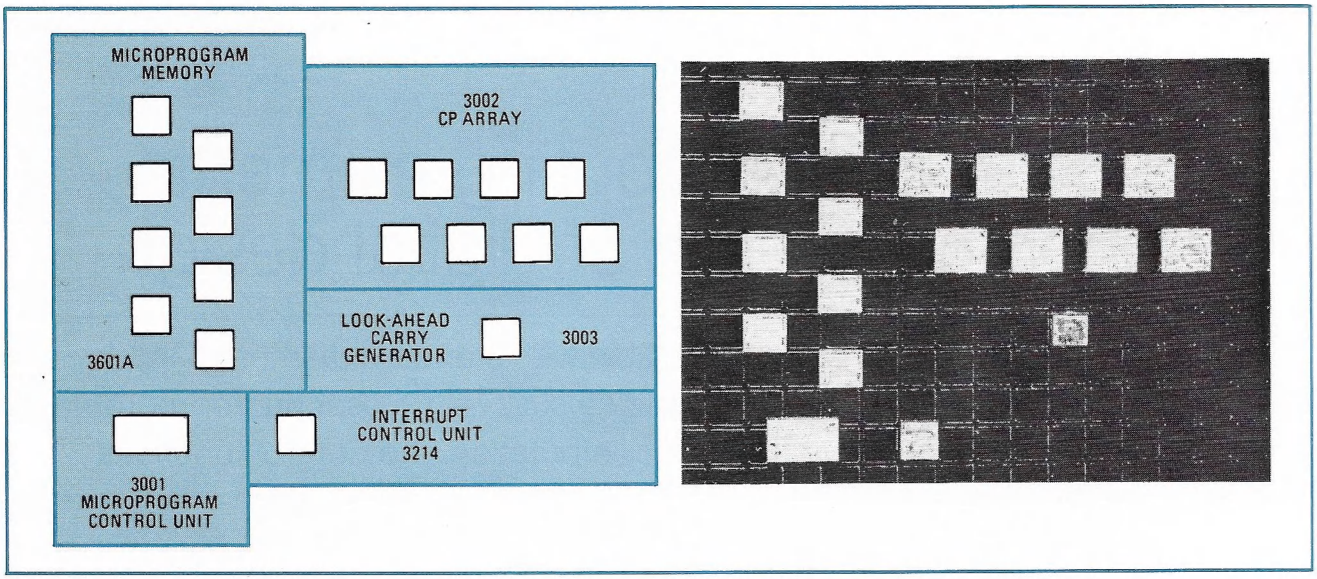
But high-performance applications are no longer beyond the reach of the microprocessor revolution. A family of computing elements has been developed using Schottky bipolar LSI technology, and it is not only faster but also far more flexible than its MOS predecessors. The new bipolar chips can be arranged in a number of different system organizations, and they are also microprogramed by the system designer to perform a number of different processing functions.[2,3]

Furthermore, systems built with this large-scale integrated circuitry are much smaller and less costly and consume less energy than equivalent designs using lower levels of transistor-transistor-logic integration. Even allowing for ancillary logic circuits, the new bipolar computing elements cut 60% to 80% off the package count in realizing most of today's designs made with small- or medium-scale-integrated TTL.

For example, an entire 16-bit processor (Fig. 1) can be assembled on one six-inch-square printed-circuit board. Such a processor is capable of typical register-to-register add cycle times of less than 125 nanoseconds. In



1. Bipolar microcomputer. Block diagram shows how to implement a typical 16-bit controller-processor with new family of bipolar computer elements. An array of eight central processing elements (CPEs) is governed by a microprogram control unit (MCU) through a separate read-only memory that carries the microinstructions for the various processing elements. This ROM may be a fast, off-the-shelf unit.



2. Family portrait. Four members of the Intel 3000 Schottky bipolar microcomputer chip set are shown here on an old transistor wafer. Interconnecting them in different combinations results in a wide range of low-cost, high-performance control and computation systems.

terms of raw computing power, that's over 15 times faster than Intel's 8080, the fastest n-channel MOS microprocessor,[4] and at double the word size.

Every LSI circuit technology confronts the chip architect and the device engineer with an assortment of design and manufacturing tradeoffs. Such factors as gate complexity, power dissipation, and propagation delay must all enter into the selection of a suitable component organization.

MOS LSI technology favors the single-chip microprocessor with its conventional control section and fixed instruction set. This judgment is based on the higher functional density and lower power dissipation realizable from MOS technology.

On the other hand, Schottky bipolar technology currently favors the multi-chip microprogramable organization. Microprogramed bipolar LSI is eminently practical because of the close match in speed that can be obtained from bipolar computing elements and memories. Additionally, since microprograming circumvents most of the complex sequential logic found within the single-chip microprocessors, bipolar die sizes are kept small and economical.

Microprogramability

A few early MOS LSI processors were microprogramed. But, while this approach offered certain benefits to the semiconductor manufacturer,[5] the devices were nearly impossible for users to microprogram. The trouble was that these processors utilized a nonstandard integrated control unit for their control and read-only memory. To test a microprogram, a user had to commit it to the mask-programed ROM element, and any change in the microprogram required a costly and time-consuming change to the program mask.

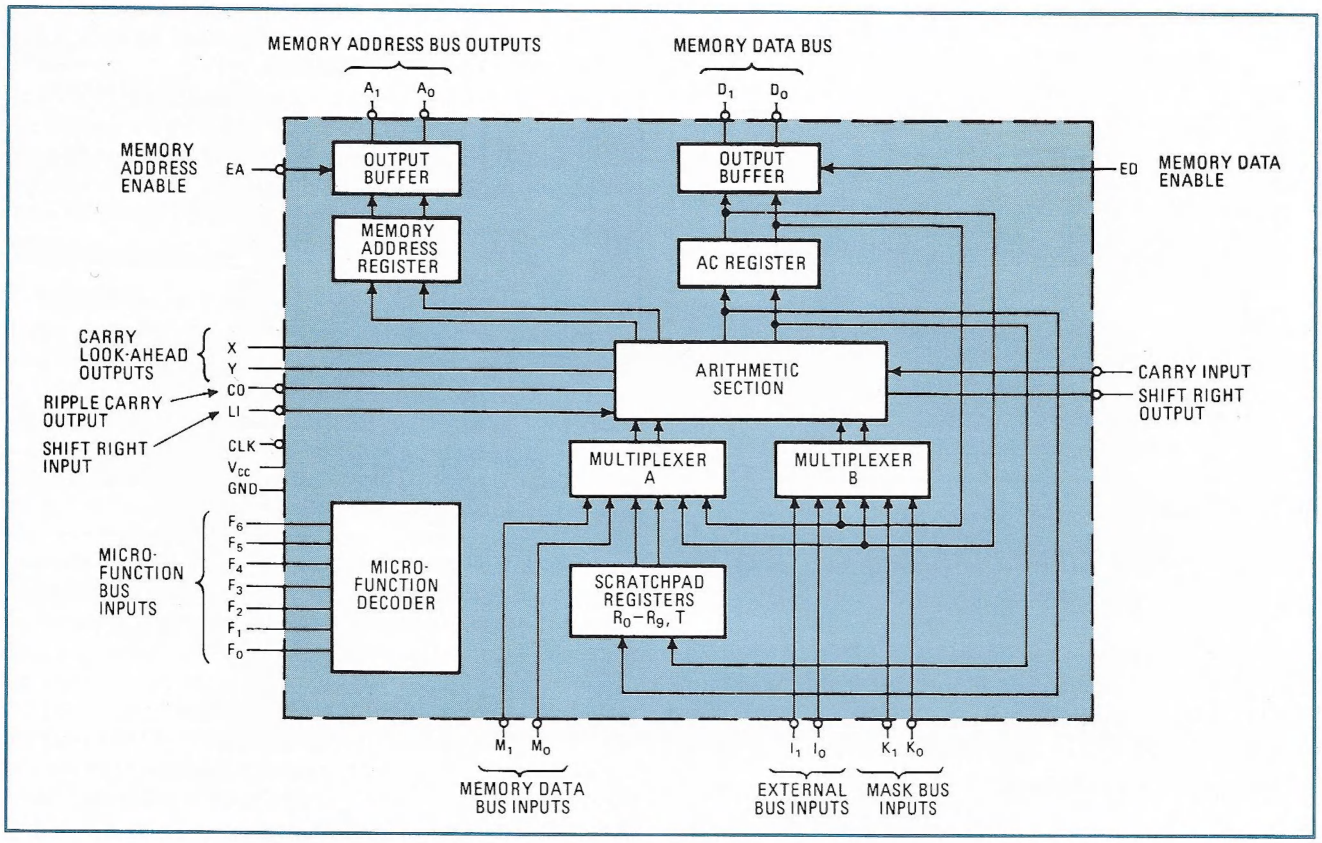
To avoid this pitfall, which would seriously downgrade the versatility and user programability of the new microcomputer set, the decision was made to utilize standard bipolar LSI memories—electrically programable and mask-programable read-only memories, as well as random-access memories, as options for storing the micro-instructions for the LSI computing elements. RAMs can hold the microprograms in developmental systems to simplify debugging. PROMs can be used to build and test prototypes or even in the production of low-volume systems. High-volume applications can commit their microprograms to compatible mask-programed ROMs. In every case the use of standard off-the-shelf components minimizes memory costs.

A family architecture

To reduce component count as far as practical, a multi-chip LSI microcomputer set must be designed as a complete, compatible family of devices. The omission of a bus or a latch or the lack of drive current can multiply the number of miscellaneous SSI and MSI packages to a dismaying extent—witness the reputedly LSI minicomputers now being offered which need over a hundred extra TTL packages on their processor boards to support one or two custom LSI devices. Successful integration should result in a minimum of extra packages, and that includes the interrupt and the input/output systems.

With this objective in mind, the Intel Schottky bipolar LSI microcomputer chip set was developed. Its two major components, the 3001 Microprogram Control Unit (MCU) and the 3002 Central Processing Element (CPE), may be combined by the digital designer with standard bipolar LSI memory to construct high-performance controller-processors with a minimum of ancillary logic.

Among the features that minimize package count and improve performance are: the multiple independent data and address busses that eliminate time multiplexing and the need for external latches; the three-state output buffers with high fanout that make bus drivers unnecessary except in the largest systems, and the separate output-enable logic that permits bidirectional busses to be formed simply by connecting inputs and outputs together.



3. Central processing element. This element contains all the circuits representing a two-bit-wide slice through a small computer's central processor. To build a processor of word width N, all that's necessary is to connect an array of N/2 CPEs together.

Each CPE represents a complete two-bit slice through the data-processing section of a computer. Several CPEs may be arrayed in parallel to form a processor of any desired word length. The MCU, which together with the microprogram memory, controls the step-by-step operation of the processor, is itself a powerful microprogramed state sequencer.

Enhancing the performance and capabilities of these two components are a number of compatible computing elements. These include a fast look-ahead carry generator, a priority interrupt unit, and a multimode latch buffer (Fig. 2). A complete summary of the first available members of this family of LSI computing elements and memories is given in the table on this page.

The cost/performance spectrum

The total flexibility of the Intel LSI computing elements is demonstrated by the broad cost/performance spectrum of the controllers and processors that can be constructed with them. These include:

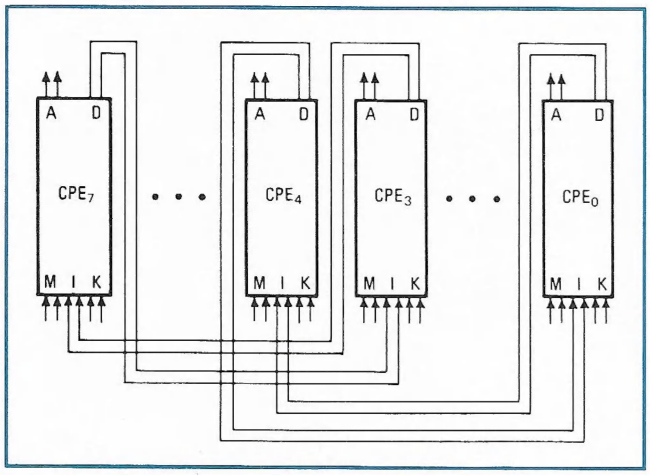
- High-speed controllers, built with a stand-alone ROM-MCU combination that sequences at up to 10 megahertz; it can be used without any CPEs as a system state controller.
- Pipelined look-ahead carry controller-processors, where the overlapped microinstruction fetch/execute cycles and fast-carry logic reduce the 16-bit add time to less than 125 nanoseconds.
- Ripple-carry controller processors (a 16-bit design adds the contents of two registers in 300 nanoseconds).
- Multiprocessors, or networks of any of the above controllers and processors, to provide computation, interrupt supervision, and peripheral control.

These configurations represent a range of microinstruction execution rates of from 3 million to 10 million instructions per second, or up to two orders of magnitude faster, for example, than p-channel microprocessors. Moreover, the increases in processor performance are achieved with relative simplicity. A ripple-carry 16-bit processor uses one MCU, eight CPEs, plus microprogram memory. One extra computing element, the 3003 Look-ahead Carry Generator, enhances the processor with fast carry. Increasing speed further by pipelining, the overlap of microinstruction fetch and execute cycles, requires a few D-type MSI flip-flops.

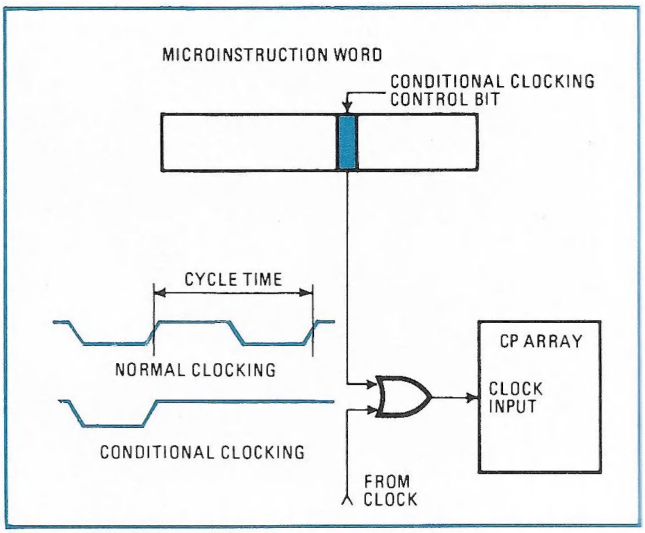
At the multiprocessor level, the microprogram memory, MCU, or CPE devices can be shared. A 16-bit processor, complete with bus control and microprogram

THE 3000 BIPOLAR LSI FAMILY

3001	Microprogram control unit
3002	Central processing element
3003	Look-ahead carry generator
3212	Multimode latch buffer
3214	Priority interrupt unit
3216	Noninverting bidirectional bus driver
3226	Inverting bidirectional bus driver
3601	256-by-4-bit programable read-only memory
3604	512-by-8-bit programable read-only memory
3301	256-by-4-bit read-only memory
3304	512-by-8-bit read-only memory



4. A byte for a byte. Used frequently in data-communications processors, a byte exchange connection exchanges high-order outputs and low-order inputs. In connection illustrated here, exchange of two highest- and lowest-order bits is shown for a 16-bit CPE array.



5. Conditional clock. This feature permits an extra bit in microinstruction to selectively control gating of clock pulse to CP array. Carry or shift data thus made available permits tests to be performed on data with fewer microinstructions.

memory, requires some 20 bipolar LSI packages and half that many small-scale ICs. In this configuration, it replaces an equivalent MSI TTL system having more than 200 packages.

Slicing up the processor

Bit slicing a processor offers a variety of device design alternatives. On the one hand, die size restrictions and pin limitations may force a slice to get by with fewer external input and output busses, and as a result, fewer independent data paths than is desirable. This has two unfortunate results.

First, the paucity of data paths causes a proliferation of external latches and multiplexers to create the address, data, and control busses found in a typical processor. Second, the inability of operands, including bit masks, to enter the slice in parallel, severely degrades performance. Multiple microcycles are required to load and operate on data in addition to the overhead cycles needed to multiplex addresses with data on a single output bus.

On the other hand, attempting to put too much logic on a single chip can also seriously degrade performance. For a given amount of power dissipation, which is determined by the selected package and limited by its cost and mechanical intricacy, the functional complexity of a device establishes its power dissipation per function ratio. As a device becomes more complex, the available power per function decreases, and the propagation delay per function increases. Clearly, longer propagation delays mean lower performance.

The organization of the 3002 CPE is a result of balancing the need for a certain number of independent busses and data paths against the current limits on die size and power dissipation.

CPEs form a processor

Each CPE (Fig. 3) carries two bits of five independent busses. The three input busses can be used in several different ways. Typically, the K-bus is used for microprogram mask or literal (constant) value input, while the other two input busses, M and I, carry data from external memory or input/output devices. D-bus outputs are connected to the CPE accumulator; A-bus outputs are connected to the CPE memory address register. As the CPEs are wired together, all the data paths, registers, and busses expand accordingly.

Certain data operations can be performed simply by connecting the busses in a particular fashion. For example, a byte exchange operation, often used in data-communications processors, may be carried out by wiring the D-bus outputs back to the I-bus inputs, exchanging the high-order outputs and low-order inputs as shown in Fig. 4. Several other discretionary shifts and rotates can be accomplished in this manner.

A sixth CPE bus, the seven-line microfunction bus, controls the internal operation of the CPE by selecting the operands and the operation to be performed. The arithmetic function section, under control of the microfunction bus decoder, performs over 40 Boolean and binary functions, including 2's complement arithmetic and logical AND, OR, NOT, and exclusive-NOR. It increments, decrements, shifts left or right, and tests for zero.

Unlike earlier MSI arithmetic-logic units, which contain many functions that are rarely used, the microfunction decoder selects only useful CPE operations. Standard carry look-ahead outputs, X and Y, are generated by the CPE for use with available look-ahead devices or the Intel 3003 Look-ahead Carry Generator. Independent carry input, carry output, shift input, and shift output lines are also available.

What's more, since the K-bus inputs are always ANDed with the B-multiplexer outputs into the arithmetic function section, a number of useful functions that in conventional MSI ALUs would require several cycles are generated in a single CPE microcycle. The type of bit masking frequently done in computer control systems can be performed with the mask supplied to the K-bus directly from the microinstruction.

Placing the K-bus in either the all-one or all-zero state will, in most cases, select or deselect the accumulator in the operation, respectively. This toggling effect of

the K-bus on the accumulator nearly doubles the CPE's repertoire of microfunctions. For instance, with the K-bus in the all-zero state, the data on the M-bus may be complemented and loaded into the CPE's accumulator. The same function selected with the K-bus in the all-one state will exclusive-NOR the data on the M-bus with the accumulator contents.

Three innovations

The power and versatility of the CPE are increased by three rather novel techniques. The first of these is the use of the carry lines and logic during non-arithmetic operations for bit testing and zero detection. The carry circuits during these operations perform a word-wide logical OR (ORing adjacent bits) of a selected result from the arithmetic section. The value of the OR, called the carry OR, is passed along the carry lines to be ORed with the result of an identical operation taking place simultaneously in the adjacent higher-order CPE.

Obviously, the presence of at least one bit in the logical 1 state will result in a true carry output from the highest-order CPE. This output, as explained later, can be used by the MCU to determine which microprogram sequence to follow. With the ability to mask any desired bit, or set of bits, via the K-bus inputs included in the carry OR, a powerful bit-testing and zero-detection facility is realized.

The second novel CPE feature is the use of three-state outputs on the shift right output (RO) and carry output (CO) lines. During a right shift operation, the CO line is placed in the high-impedance (Z) state, and the shift data is active on the RO line. In all other CPE operations, the RO line is placed in the Z state, and the carry data is active on the CO line. This permits the CO and RO lines to be tied together and sent as a single rail input to the MCU for testing and branching. Left shift operations utilize the carry lines, rather than the shift lines, to propagate data.

The third novel CPE capability, called conditional clocking, saves microcode and microcycles by reducing the number of microinstructions required to perform a given test. One extra bit is used in the microinstruction to selectively control the gating of the clock pulse to the central processor (CP) array. Momentarily freezing the clock (Fig. 5) permits the CPE microfunction to be performed, but stops the results from being clocked into the specified registers. The carry or shift data that results from the operation is available because the arithmetic section is combinatorial, rather than sequential. The data can be used as a jump condition by the MCU and in this way permits a variety of nondestructive tests to be performed on register data.

A good example of the over-all capability of the CPE is stack pointer or program counter maintenance. Usually this operation requires four microprocessor cycles: fetch the register data, send it to the memory address register, increment the value, and store the result back in the specified register. But the CPE can do it in one microcycle, using any one of its 11 scratchpad registers as the stack pointer or program counter. The desired address is gated through the arithmetic section to the memory address register, and the memory cycle is initiated. Simultaneously, over a separate data path, the address value is incremented by the arithmetic section and sent back to the scratchpad.

By itself the CP array is incomplete as a microprocessor. The arithmetic, logic, and register functions need to be controlled in some orderly fashion, and this is the function of the MCU.

Microprograming technology

- **Microprogram:** A type of program that directly controls the operation of each functional element in a microprocessor.
- **Microinstruction:** A bit pattern that is stored in a microprogram memory word and specifies the operation of the individual LSI computing elements and related subunits, such as main memory and input/output interfaces.
- **Microinstruction sequence:** The series of microinstructions that the microprogram control unit (MCU) selects from the microprogram to execute a single macroinstruction or control command. Microinstruction sequences can be shared by several macroinstructions.
- **Macroinstruction:** Either a conventional computer instruction (e.g. ADD MEMORY TO REGISTER, INCREMENT, and SKIP, etc.) or device controller command (e.g., SEEK, READ, etc.).

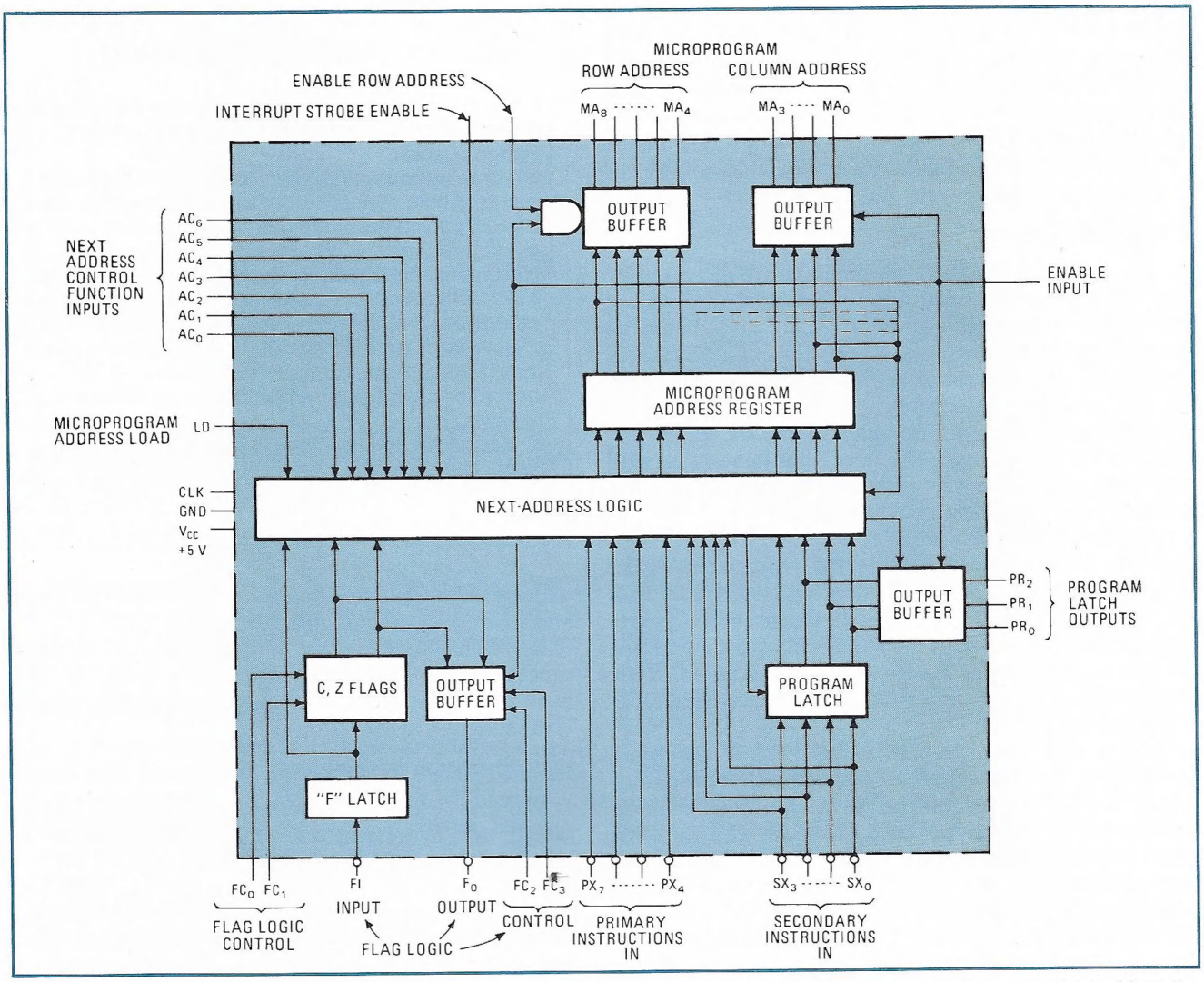
Microprogram control

The classic form of microprogram control incorporates a next-address field in each microinstruction—any other approach would require some type of program counter. To simplify its logic, the MCU (Fig. 6) uses the classic approach and requires address control information from each microinstruction. This information is not, however, simply the next microprogram address. Rather, it is a highly encoded specification of the next address and one of a set of conditional tests on the MCU bus inputs and registers.

The next-address logic and address control functions of the MCU are based on a unique scheme of memory addressing. Microprogram addresses are organized as a two-dimensional array or matrix, as shown in Fig. 7. Unlike in ordinary memory, which has linearly sequenced addresses, each microinstruction is pinpointed by its row and column address in the matrix. The 9-bit microprogram address specifies the row address in the upper 5 bits and the column address in the lower 4 bits. The matrix can therefore contain up to 32 row addresses and 16 column addresses for a total of 512 microinstruction addresses.

The next-address logic of the MCU makes extensive use of this addressing scheme. For example, from a particular row or column address, it is possible to jump either unconditionally to any other location in that row or column or conditionally to other specified locations, all in one operation. For a given location in the matrix there is a fixed subset of microprogram addresses that may be selected as the next address. These are referred to as a jump set, and each type of MCU address control jump function has a jump set associated with it.

Incorporating a jump operation in every microinstruction improves performance by allowing process-



6. Microprogram control unit. The MCU's two major control functions include controlling the sequence of microprograms fetched from the microprogram memory, and keeping track of the carry inputs and outputs of the CP array by means of the flag logic control.

ing functions to be executed in parallel with program branches. Reductions in microcode are also obtained because common microprogram sequences can be shared without the time-space penalty usually incurred by conditional branching.

Independently controlled flag logic in the MCU is available for latching and controlling the value of the carry and shift inputs to the CP array. Two flags, called C and Z, are used to save the state of the flag input line. Under microprogram control, the flag logic simultaneously sets the state of the flag output line, forcing the line to logical 0, logical 1, or the value of the C or Z flag.

The jump decisions are made by the next-address logic on the basis of: the MCU's current microprogram address; the address control function on the accumulator inputs; and the data that's on the macroinstruction (X) bus or in the program latch or in the flags. Jump decisions may also be based on the instantaneous state of the flag input line without loading the value in one of the flags. This feature eliminates many extra microinstructions that would be required if only the flag flip-flop could be tested.

Microinstruction sequences are normally selected by the operation codes (op codes) supplied by the microinstructions, such as control commands or user instructions in main memory. The MCU decodes these commands by using their bit patterns to determine which is to be the next microprogram address. Each decoding results in a 16-way program branch to the desired microinstruction sequence.

Cracking the op codes

For instance, the MCU can be microprogramed to directly decode conventional 8-bit op codes. In these op codes the upper 4 bits specify one of up to 16 instruction classes or address modes, such as register, indirect, or indexed. The remaining bits specify the particular subclass such as ADD, SKIP IF ZERO, and so on. If a set of op codes is required to be in a different format, as may occur in a full emulation, an external pre-decoder, such as ROM, can be used in series with the X-bus to reformat the data for the MCU.

In rigorous decoding situations where speed or space is critical, the full 8-bit macroinstruction bus can be used for a single 256-way branch. Pulling down the load line of the MCU forces the 8 bits of data on the X-bus (typically generated by a predecoder) directly into the microprogram address register.

The data thus directly determines the next microprogram address which should be the start of the desired microprogram sequence. The load line may also be used by external logic to force the MCU, at power-up, into the system re-initialization sequence.

From time to time, a microprocessor must examine the state of its interrupt system to determine whether an interrupt is pending. If one is, the processor must suspend its normal execution sequence and enter an interrupt sequence in the microprogram. This requirement is handled by the MCU in a simple but elegant manner.

When the microprogram flows through address row 0 and column 15, the interrupt strobe enable line of the MCU is raised. The interrupt system, an Intel 3214 Interrupt Control Unit, responds by disabling the row address outputs of the MCU via the enable row address line, and by forcing the row entry address of the microprogram interrupt sequence onto the row address bus. The operation is normally performed just before the macroinstruction fetch cycle, so that a macroprogram is interrupted between, not during, macroinstructions.

The 9-bit microprogram address register and address bus of the MCU directly address 512 microinstructions. This is about twice as many as required by the typical 16-bit disk-controller or central processor.

Moreover, multiple 512 microinstruction memory planes can easily be implemented simply by adding an extra address bit to the microinstruction each time the number of extra planes is doubled. Incidentally, as the number of bits in the microinstruction is increased, speed is not reduced. The additional planes also permit program jumps to take place in three address dimensions (Fig. 8) instead of two.

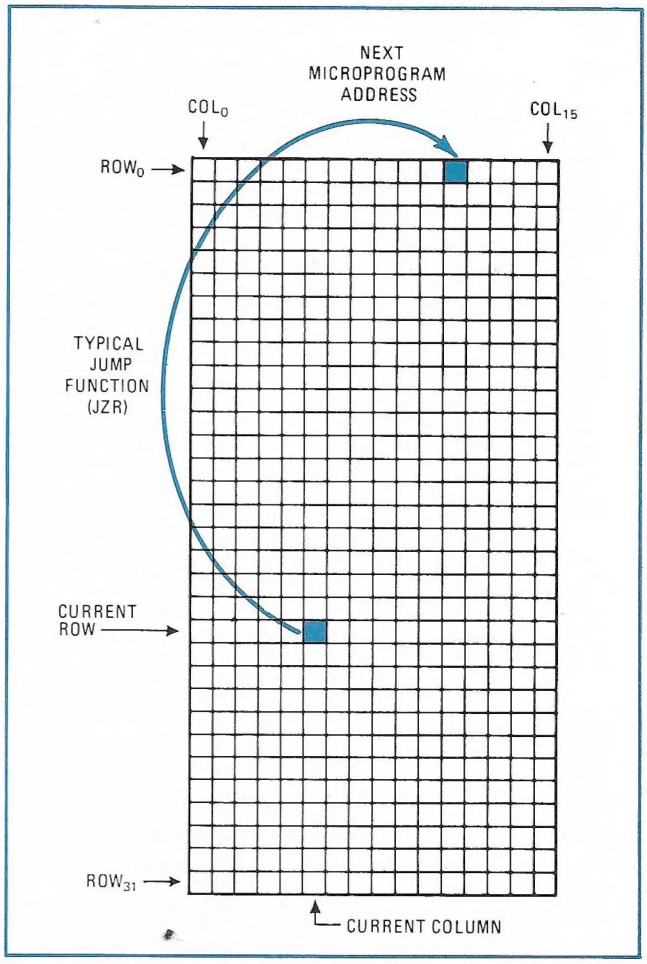
Because of the tremendous design flexibility offered by the Intel computing elements, it is impossible to describe every microinstruction format exactly. But generally speaking, the formats all derive from the one in Fig. 9. The minimum width is 18 bits: 7 bits for the address control functions, plus 4 bits for the flag logic control; plus 7 bits for the CPE microfunction control.

More bits can be added to the microinstruction format to provide such functions as mask field input to the CP array, external memory control, conditional clocking, and so on. Allocation of these bits is left to the designer who organizes the system. He is free to trade off memory costs, support logic, and microinstruction cycles to meet his cost/performance objectives.

Configuring a processor

The processor organization of Fig. 1 may be varied to enhance speed, reduce component count, or increase data-processing capability. As mentioned earlier, one widely applicable technique for maximizing a processor's performance is called pipelining. To pipeline a microprocessor, a group of D-type flip-flops is hung on the microprogram memory outputs (excluding the address control field) to buffer the current microinstruction and so allow the MCU to overlap the fetch of the next microinstruction with the execution of the current one. If fast carry logic is also used, the microinstruction cycle time is typically less than 125 nanoseconds.

Although almost any number of CPEs may be arrayed (up to 320 bits without buffering, to be exact), a system



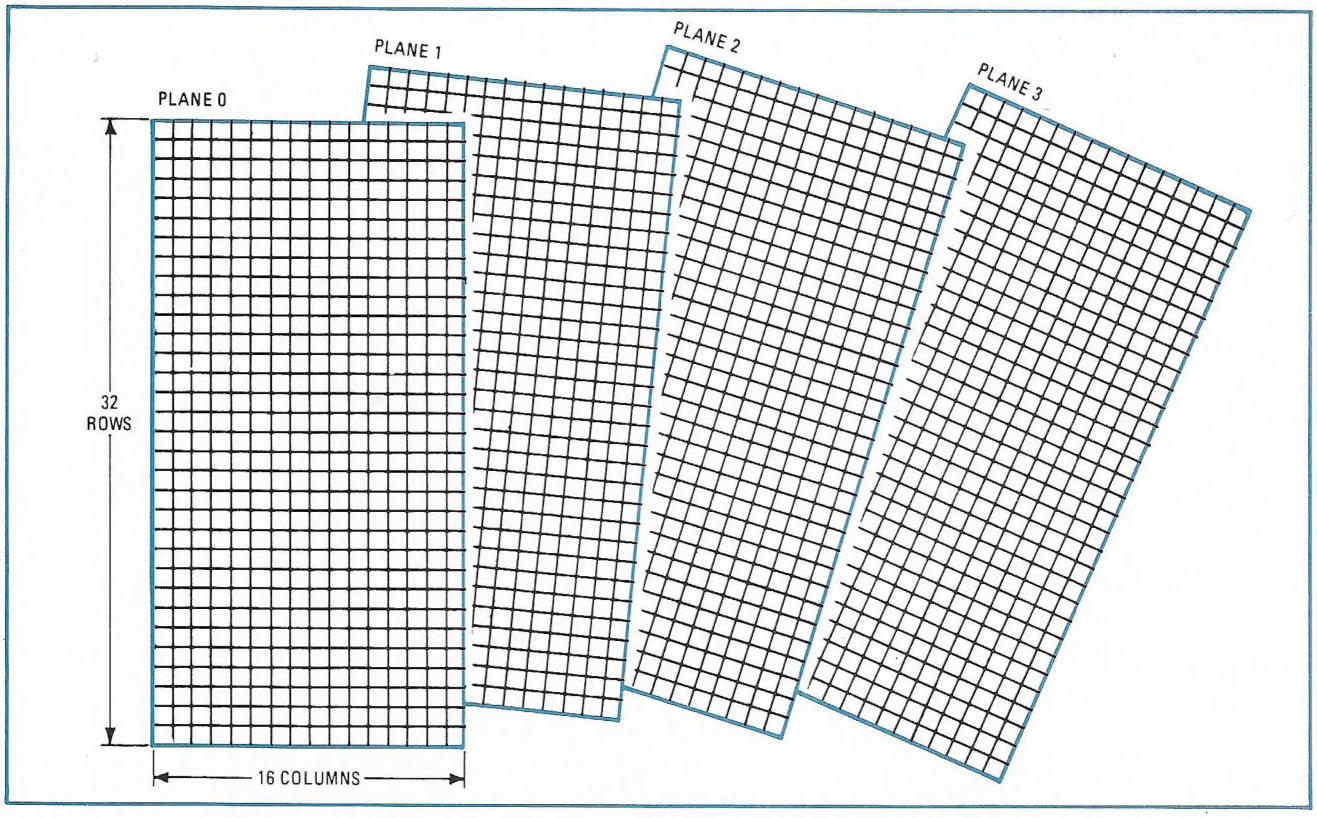
7. Jump function. Microprogram addresses, represented here by boxes, are organized as a two-dimensional array or matrix. Each microinstruction contains a jump operation field that specifies the next microprogram address. Shown here is a typical jump operation from location (row 20-column 5) to (row 0-column 11).

using only a few CPEs may not have a CP array address bus wide enough for the size of main memory required. A good example is the 8-bit processor that needs a 16-bit address bus. Such problems are easily solved. In this case the CP array delivers one byte of the address on its memory address bus outputs and the other byte on its data bus outputs. The tradeoff is that two microcycles are required to send the address to memory.

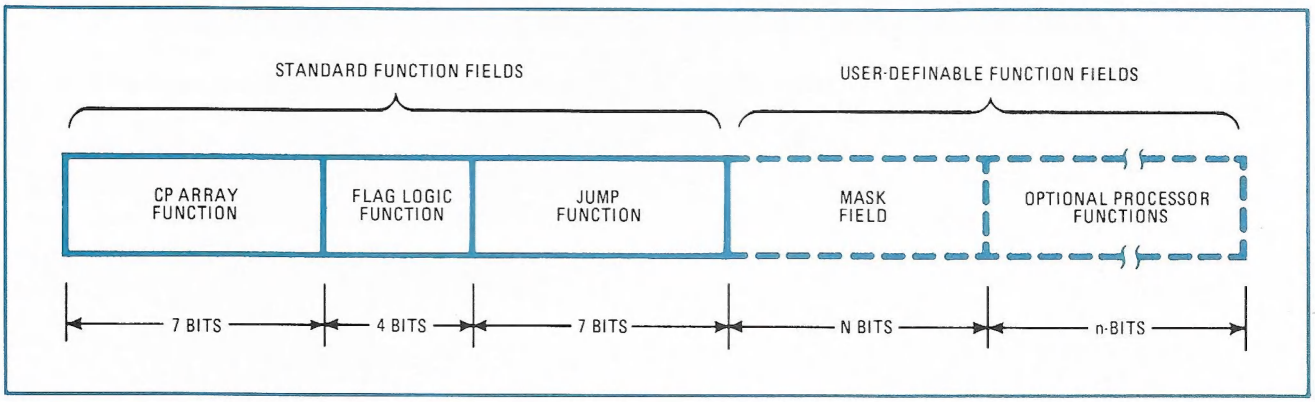
Another configuration insures rapid servicing of interrupts by combining two CP arrays with one MCU. All of the register contents associated with the interrupted routine are held in one array, while the interrupt service routine is executed on the second CP array. Normal program execution resumes when the MCU regains control of the first CP array.

Multiprocessing is another way to obtaining high performance. Several satellite MCU-CP arrays can share a common main memory—an arrangement that could be used quite effectively in a multi-terminal information system. In a busy system, memory bus conflicts might increase the average processor cycle time, but the speed of the Schottky bipolar computing elements is so great that such delays would be invisible to the users.

As a final example, consider the costly and complex



8. Multiplanar addressing. If required, multiple planes, each carrying 512 microinstructions, can be implemented without any sacrifice in the speed of the system. Moreover, jumping is possible in three address dimensions with such a configuration.



9. Microinstruction format. Only a generalized microinstruction format can be shown since allocation of bits for the mask field and optional processor functions depends on the wishes of the designer and the tradeoffs he decides to make.

controllers available for the inexpensive diskette or floppy disk. The relatively high data-transfer rates and the processing demands of the soft-sectored diskette put this application just beyond the reach of the fastest MOS microprocessors. Existing controllers contain upwards of 250 TTL packages and can only work with one type of media-formatting scheme.

In contrast, a universal diskette controller, adaptable to a variety of formatting schemes through microprograming, requires less than 20 bipolar LSI components, including the MCU, CPEs, and memories. The over-all diskette controller package count is reduced by two thirds.

Intel's Schottky bipolar computing elements are the first LSI technique endowed with the computing and control power of discrete high-speed TTL circuits. In addition, they give the digital designer a degree of flexibility, simplicity, and economy never before available to him. By all indications, bipolar LSI computing elements are destined to become the standard for high-speed design in the next generation of computation and control systems. □

REFERENCES
1. M.E. Hoff, Jr., "New LSI Components," 8th IEEE Computer Society International Conference Digest, 1972, pp. 141–143.
2. S.S. Husson, "Microprogramming: Principles and Practice," Prentice Hall Inc., 1970.
3. P.M. Davies, "Readings in Microprogramming," IBM Systems Journal, Vol. 11, 1972, pp. 16–40.
4. M. Shima and F. Faggin, "In switching to n-MOS microprocessor gets 2-microsecond cycle time," *Electronics*, April 18, 1974, pp. 95–100.
5. G. Reyling, Jr., "Considerations in Choosing a Microprogrammable Bit-Slice Architecture," Computer, July, 1974, pp. 26–29.

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 246-7501